INSA | INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

RAPPORT B.E. RÉSEAU

DURAND-SIMONNET Éole
PIOCHE Océane
3 MIC-D

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

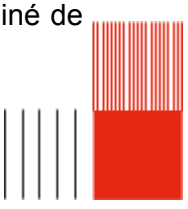
Lors de ce bureau d'étude, nous sommes arrivées jusqu'à l'étape de l'établissement de la phase de connexion de la v4.1.
Jusque là, nous avons réussi à bien avancer sans trop de difficulté pendant chaque séance de TP.

Pour l'implémentation de la version 1, nous n'avions pas vraiment de choix décisif à faire. Nous avons codé à l'aide du cours et de ce que nous avions vu en séance de TD, pour créer une version de MICTCP incluant une phase de transfert de données sans garantie de fiabilité. Cela nous a pris toute la première séance de TP.

Pour l'implémentation de la version 2, nous avons décidé de suivre la version finale de l'algorithme "Stop and Wait" du cours d'Introduction au Réseau. Cela nous a pris environ la seconde séance de TP.

Pour l'implémentation de la version 3, nous avons dû inclure le mécanisme de reprise de pertes dans notre code. Pour cela, il a fallu modifier la fonction *mic_tcp_send*. Dans le code final, nous avons décidé de fixer le taux de pertes à 0,5.
Dans la boucle while, on cherche à attendre un acquittement du paquet venant d'être envoyé. Si on ne reçoit pas l'acquittement, on regarde si la perte est tolérable ou non en fonction du taux de perte fixé. Si celui-ci est tolérable, alors on décide de "perdre" le paquet. Dans le cas contraire, il faudra renvoyer ce dernier. Si on reçoit un acquittement, on vérifie alors que le paquet reçu correspond bien à celui envoyé. Si ce n'est pas le cas, alors on regarde si la perte de ce paquet est tolérable en comparant avec le taux de perte. Si c'est le cas, on décide de "perdre" le paquet, sinon il faudra le renvoyer. Nous avons fini d'implémenter la version 3 lors de la troisième séance de TP.

Pour l'implémentation de la version 4.1, il fallait étendre la phase de transfert de sorte à inclure un phase d'établissement de connexion, ainsi qu'une garantie de fiabilité partielle via un mécanisme de reprise de pertes de type "Stop and Wait" dont le % de pertes admissibles sera négocié durant la phase d'établissement de connexion. Nous avons donc d'abord modifié la fonction *mic_tcp_accept* de sorte à appeler cette fonction tant que la connexion n'est pas établie.
Ensuite, dans la fonction *mic_tcp_connect*, on initialise le PDU SYN du paquet à émettre, puis on cherche à savoir si on reçoit bien le SYN ACK que l'on attend. Si le PDU reçu avant l'expiration du timer est bien le SYN ACK, alors on déclare avoir reçu ce dernier pour pouvoir sortir de la boucle while. Il nous reste ensuite à initialiser l'ACK à envoyer, puis finir l'envoi pour établir la connexion.
Dans la fonction *process_received_PDU*, nous traitons le cas de réception d'un SYN, d'envoi d'un SYN ACK, de réception d'un ACK et de réception d'une data. Nous avons terminé de



coder cette partie-là lors de la dernière séance de TP. Par manque de temps nous n'avions pas pu avancer plus loin.

Ainsi, en travaillant de manière régulière lors des séances de TD et de TP, nous avons réussi à avancer sur ce BE d'une façon efficace. Ce projet s'est réalisé avec une bonne dynamique au sein du groupe. Nous avons décidé de nous arrêter à la v4.1 car il nous manquait du temps pour pouvoir terminer la v4.2.